On the selection of the globally optimal prototype subset for nearest-neighbor classification

INFORMS Journal on Computing
Vol. 19, No. 3, Summer 2007, pp. 470–479
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1903 | 0470

DOI 10.1287/ijoc.1060.0183

On the Selection of the Globally Optimal Prototype Subset for Nearest-Neighbor Classification

Emilio Carrizosa
Facultad de Matemáticas, Universidad de Sevilla, 41012 Sevilla, Spain, ecarrizosa@us.es

Belén Martín-Barragán
Departamento de Estadística, Universidad Carlos III de Madrid, 28903 Getafe, Madrid, Spain, belen.martin@uc3m.es

Frank Plastria
MOSI–Department of Mathematics, Operational Research, Statistics and Information Systems for Management, Vrije Universiteit Brussel, B-1050 Brussel, Belgium, frank.plastria@vub.ac.be

Dolores Romero Morales
Saïd Business School, University of Oxford, Oxford OX1 1HP, United Kingdom, dolores.romero-morales@sbs.ox.ac.uk

The nearest-neighbor classifier has been shown to be a powerful tool for multiclass classification. We explore both theoretical properties and empirical behavior of a variant method, in which the nearest-neighbor rule is applied to a reduced set of prototypes. This set is selected a priori by fixing its cardinality and minimizing the empirical misclassification cost. In this way we alleviate the two serious drawbacks of the nearest-neighbor method: high storage requirements and time-consuming queries. Finding this reduced set is shown to be NP-hard. We provide mixed integer programming (MIP) formulations, which are theoretically compared and solved by a standard MIP solver for small problem instances. We show that the classifiers derived from these formulations are comparable to benchmark procedures. We solve large problem instances by a metaheuristic that yields good classification rules in reasonable time. Additional experiments indicate that prototype-based nearest-neighbor classifiers remain quite stable in the presence of missing values.

Key words: classification; optimal prototype subset; nearest neighbor; dissimilarities; integer programming; variable neighborhood search; missing values
History: Accepted by Amit Basu, Area Editor for Knowledge and Data Management; received July 2003; revised May 2005, October 2005; accepted February 2006. Published online in *Articles in Advance* July 20, 2007.

1. Introduction

A classification problem has a database with objects belonging to $|C|$ different classes, and one wants to derive a *classification rule*, i.e., a procedure that labels every future entry as a member of one of the $|C|$ existing classes. Classification procedures are of two types: *parametric* and *nonparametric*. Parametric procedures assume that each object from class $c \in C$ is associated with a random vector with known distribution, perhaps up to some parameters to be estimated, (e.g. data are multivariate normal vectors with unknown mean μ_c and covariance matrix Σ_c), and primarily use of statistics (see e.g. McLachlan 1992) For complex databases with no evident distributional assumptions on the data (typical for databases with both quantitative and qualitative variables), nonparametric methods, such as the one described in this paper, are needed.

There has been increasing interest in deriving (nonparametric) classification rules via mathematical programming, requiring, for each object i, a vector v^i of n numerical variables. This assumes variables to be ratio-scaled, rather than nominal or ordinal. Moreover, no null values are allowed, which excludes its direct use for cases in which some measures are missing or simply do not apply; see Cristianini and Shawe-Taylor (2000), Freed and Glover (1981), Gehrlein (1986), Gochet et al. (1997), and Mangasarian (1994).

A more flexible methodology, which requires only knowledge of a metric (or, as discussed in Section 2.1, a dissimilarity), is the nearest-neighbor (NN) method (Cover and Hart 1967, Dasarathy 1991, Devroye et al. 1996, Hastie et al. 2001), which provides excellent results (King et al. 1995, Pekalska et al. 2006). In NN methods, for each new entry i the distances (or dissimilarities) $d(i, j)$ to some objects j in the database (called *prototypes*) are computed, and i is classified according to these distances. In particular, in the classical NN method, (Cover and Hart 1967), all objects are prototypes, and i is classified as member of the class c^* to which its closest prototype j^* (satisfying $d(i, j^*) \leq d(i, j)\ \forall j$) belongs.

A generalization of the NN method is the k-NN method (Devroye et al. 1996), which classifies each i to the class most frequently found among the k prototypes closest to i. In particular, NN is the special case of k-NN for $k = 1$.

These classification rules, however, require distances to be calculated to all data in the database for each new entry, involving high storage and time resources, making it impractical to perform online queries, so several variants have been proposed (Bennett and Willemain 2004, Bezdek and Kuncheva 2001, Dasarathy 1991, Devroye et al. 1996, Geva and Sitte 1991, Hart 1968, Kuncheva 1997, Kuncheva and Bezdek 1998). Most proposals differ in the way they attempt to provide heuristically a set of prototypes of small size and low misclassification cost (Bezdek and Kuncheva 2001). An extreme case is the condensed nearest-neighbor (CNN) rule (Hart 1968), in which the full database I is replaced by a so-called minimal consistent subset, a smallest subset S of records such that, if the NN classifier is used with S (instead of I) as the set of prototypes, all objects in I are correctly classified.

Since the size of a minimal consistent subset is unpredictable and might still be too large, several procedures have been suggested to reduce its size. Although such procedures do not necessarily correctly classify all items in the database (i.e., they are not consistent), they may have similar or even better behavior to predict class membership on future entries because they may reduce the possible overfitting suffered by CNN (Brighton and Mellish 2002, Lipowezky 1998).

In Bezdek and Kuncheva (2001) a number of procedures of this type are classified according to three different issues:

- The prototype design: the prototypes can either be selected from I or be constructed (e.g. by considering centroids for numerical variables)
- The use of labels, i.e., whether the class labels of the sample data are used or not to select/construct the prototypes
- The control on the size of the set of prototypes, i.e., whether the number of prototypes is specified in advance or is automatically determined by the algorithm.

We propose a method in which a set of prototypes of pre-specified cardinality p is sought, minimizing an empirical misclassification cost. As prototype design, we assume that prototypes are to be chosen from a given set, not necessarily equal to the set of available data. Hence, we give the highest freedom in this issue. Labels from sample data are used, so all existing information is taken into account. Finally, the user chooses the number of prototypes, and therefore fully controls the query times, which are critical when the computation of dissimilarities is costly. Indeed, the effort needed to classify a new entry is directly proportional to p and may therefore guide the choice of p.

We restrict ourselves to the classification rule based on the closest distance, so our method is as a variant of NN. However, our results may directly be extended to considering the k closest distances, $k \geq 1$, in the classification procedure, leading to a variant of the k-NN method.

The model is introduced in Section 2 and shown to be $\mathcal{NP}$-hard. In Section 3, two mixed integer programming (MIP) formulations are proposed and theoretically compared. Numerical results are given in Section 4. When the optimization problems are solved exactly (with a standard MIP solver) the behavior of the classification rule is promising, but at the price of enormous preprocessing times. For this reason, a heuristic procedure is also proposed, and its quality and speed is explored. The rules obtained with this heuristic procedure have similar behavior on testing samples as the optimal ones. The method remains quite stable in the presence of missing values. Some concluding remarks and possible extensions are in Section 5.

2. The Model

2.1. Classification Rules

A key concept in NN-based classification methods is the concept of distance, or, more generally, *dissimilarity*. A dissimilarity on a set J is a function $d\colon J \times J \to \mathbb{R} \cup \{+\infty\}$, satisfying

$$d(u, v) \geq 0, \quad \forall\, u, v \in J \tag{1}$$

$$d(u, u) = 0, \quad \forall\, u \in J. \tag{2}$$

When the set J is (a subset of) the n-dimensional space $\mathbb{R}^n$, the most popular dissimilarities are those derived from metrics, such as the (weighted) Euclidean or the Mahalanobis distance. See Plastria (1995, 2001) for further details, extensions, and modelling aspects.

However, not all interesting dissimilarity measures correspond to metrics. In a typical example (Kaufman and Rousseeuw 1990) J is a finite set of the n-dimensional space, but for some objects some of its coordinates cannot be used (because they are missing, or strongly suspected to be wrong). In that case, denoting, for each $u \in \mathbb{R}^n$, by $D(u)$ the set of coordinates of u that can be used, we can extend the definition of Euclidean distance to the dissimilarity (not necessarily metric)

$$d(u, v) = \begin{cases} \left(\displaystyle\sum_{j \in D(u) \cap D(v)} \frac{\omega_j}{|D(u) \cap D(v)|} (u_j - v_j)^2 \right)^{1/2} \\ \qquad \text{if } D(u) \cap D(v) \neq \varnothing \\ +\infty \quad \text{otherwise,} \end{cases} \tag{3}$$

for given weights $\omega_1, \ldots, \omega_n$.

Natural definitions abound of dissimilarities for sets J for which a metric is neither feasible nor recommended; see Kaufman and Rousseeuw (1990), where some dissimilarities are defined for sets J of objects for which n variables are measured, some quantitative, some ordinal or nominal, and nulls, as in (3), exist. Other methods for deriving dissimilarities can be found in protein/amino-acid alignment in bioinformatics (Altschul et al. 1994, 1990; Pearson and Lipman 1998), or fuzzy analysis, usually as the complement to 1 of a *fuzzy similarity relation* (Zimmermann 1991). In Yang and Shih (2001) set J is a set of portraits of people from three different families.

Let J be a finite set of objects with a dissimilarity d defined on it, partitioned into $|C|$ classes J_c ($c \in C$). A *classification rule* is a function φ: $J \to C$ that associates with each object $s \in J$ a class that might be its correct class or not. We consider only classification rules based on selecting *prototypes* for the different classes in C as follows.

For each class $c \in C$ we are given a nonempty set $R_c \subset J$ of *prototype candidates* of class c. We denote the full set of candidates as R and assume that the sets R_c produce a partition of R, i.e. $\bigcup_{c\in C} R_c = R$ and $R_c \cap R_{c'} = \varnothing$, $c \neq c'$.

Some nonempty $S \subset R$, the set of prototypes, is to be chosen. For a given S, let φ_S be the S-based NN classification rule, namely the classification rule that labels each $i \in J$ with the (known) label of the prototype in S closest (i.e., least dissimilar) to i. In other words, if $d(i, S \cap R_c)$ denotes $d(i, S \cap R_c) = \min_{j\in S\cap R_c} d(i, j)$, then let

$$\varphi_S(i) = \arg\min_{c\in C} d(i, S \cap R_c), \tag{4}$$

if this minimum is attained at a single c. In case of ties, a least-dissimilar c must be chosen as $\varphi_S(i)$, as detailed in the next section.

2.2. Performance Measure

For each $i \in J$ let $c(i) \in C$ denote the class to which i belongs. In general, misclassification errors, i.e., objects i with $\varphi_S(i) \neq c(i)$, will exist. Since not all misclassification errors are equally important, we assume we know for each $c, c' \in C$, the misclassification penalty $r(c' \mid c) \geq 0$, associated with each object of class c labelled as a member of class c'. Let $r(c \mid c) = 0$, i.e., the cost of correct classification is zero.

A particular but important case is when all wrong classifications contribute the same cost,

$$r(c' \mid c) = \begin{cases} r_c & \text{if } c' \neq c \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where, for each $c \in C$, $r_c > 0$. We will call this the *uniform case*. Moreover, when all r_c are equal, say, to unity, one obtains the *binary case*

$$r(c' \mid c) = \begin{cases} 1 & \text{if } c' \neq c \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

which counts the number of misclassified objects (Mangasarian 1994). Our method accommodates cost structures more general than (6), e.g. as needed in Carrizosa et al. (2005), who construct a classifier for a cancer-diagnosis problem where different misclassification types naturally imply different misclassification costs.

Assignment rules should also be defined in case of ties: assignment will be done to the least dissimilar prototype, and ties will be broken randomly or by some user-defined procedure.

To compute the cost associated with a classifier, we use a worst-case approach. Since R is a finite set, we can arbitrarily sort its labels, giving a strict total order $\prec$ on R. For each $i \in J$, let $\prec_i$ be the strict total order on R yielding the assignment for i: For any $j_1, j_2 \in R$, $j_1 \prec_i j_2$ iff one of the following holds:

- $d(i, j_1) < d(i, j_2)$
- $d(i, j_1) = d(i, j_2)$, $r(c(j_1) \mid c(i)) > r(c(j_2) \mid c(i))$
- $d(i, j_1) = d(i, j_2)$, $r(c(j_1) \mid c(i)) = r(c(j_2) \mid c(i))$, $j_1 \prec j_2$.

Since $\prec_i$ is a strict total order on R, it is also a strict total order on any nonempty $S \subset R$. Hence, $\{j \in S$: no $j' \in S$ satisfies $j' \prec_i j\}$ is a singleton and its class defines $\varphi_S(i)$. This definition is consistent with (4) and extends it to the case of ties in dissimilarities.

With this cost structure, the total cost $\pi_J(S)$ *within* J of the S-based NN classification rule φ_S is $\pi_J(S) = \sum_{i\in J} r(\varphi_S(i) \mid c(i))$, which is a performance measure of the classification rule.

Evaluation of $\pi_J(\cdot)$ implicitly requires complete knowledge of the member class of each object in J. In practice, $c(i)$ will be known only for objects in a set $I \subset J$, called the *training set*. Hence, the definition of π_J is of limited use since it cannot be calculated. However, if we assume that the training set I has been obtained by a sampling in J, (unbiased) estimators $\widehat{\pi}_J$ of π_J can be used as surrogates; see Cochran (1977) for an introduction to sampling and statistical estimation strategies.

Indeed, suppose that I has been obtained after sampling in J, using a sampling design such that, for any $c \in C$, any object i in class c is included in the sample with probability $pr(c) > 0$. For instance, if I is obtained by random sampling without replacement of size s, then

$$pr(c) = \frac{s}{|J|} \quad \forall\, c \in C. \tag{7}$$

With stratified random sampling, a random sample without replacement I_c of size $s(c)$ is drawn from each $c \in C$, yielding (Cochran 1977) $pr(c) = s(c)/|c|$. In any

case, an unbiased estimator for $\pi_I(S)$ is the Horwitz-Thompson estimator (Thompson 2002)

$$\widehat{\pi_I(S)} = \sum_{i \in I} r(\varphi_S(i) \mid c(i))/pr(c(i)).$$

We assume that, as in (7), all probabilities $pr(c)$ are equal, so $\widehat{\pi_I(S)}$ is proportional to the total cost $\pi_I(S)$ *within the training sample* I,

$$\pi_I(S) = \sum_{i \in I} r(\varphi_S(i) \mid c(i)), \tag{8}$$

called the empirical classification cost. Thus, for the particular cost structure (6), the empirical classification cost is simply the number of misclassified objects in the training sample I.

Given an integer p, $|C| \le p \le |R|$, we determine the classification rule φ_S with minimal empirical cost, measured as (8), such that S is a subset of R, with cardinality p, and at least one prototype from each class c is included, i.e. $S \cap R_c \neq \varnothing$, $\forall c$. We call this the *optimal p-prototypes nearest-neighbor* (p-PNN) model. If $R = I$, i.e., admitting the full training set as candidates to prototypes, then $|I|$-PNN is NN.

2.3. Complexity

In this section we prove that finding a p-PNN rule is $\mathcal{NP}$-hard and that the problem remains $\mathcal{NP}$-hard even when restricted to two-class case, the set of candidates to prototypes coinciding with the training set, the dissimilarity is a metric and the misclassification costs are uniformly equal to one. We formalize this in the following decision problem

PERFECT CLASSIFICATION: Given a number p and a finite set I, partitioned into two subsets I_1, I_2 and equipped with a metric d, does there exist a subset S of I of cardinality p such that the corresponding classification rule classifies all elements of I correctly?

Proposition 1. *PERFECT CLASSIFICATION is an $\mathcal{NP}$-complete problem.*

Proof. Our starting point is the following $\mathcal{NP}$-complete problem (Garey and Johnson 1979):

DOMINATING SET: Given a graph (V, E) and a positive number $l \le |V|$, does there exist an l-dominating vertex set? An l-dominating vertex set is a subset $V' \subseteq V$ with $|V'| \le l$ and such that all vertices in $V \backslash V'$ are adjacent to V'.

For instance, consider the graph (V, E) in Figure 1. $\{v_4, v_5, v_6, v_8\}$ is a 4-dominating vertex set for (V, S), whereas $\{v_4, v_5, v_6\}$ is not a 3-dominating vertex set, since it contains no vertex adjacent to v_9.

Given an instance of DOMINATING SET, we construct an instance of PERFECT CLASSIFICATION as follows. Let $I_1 = V$ and $I_2 = \{w\}$ where w is an arbitrary object not in V and $p = l + 1$. The dissimilarities are defined by the shortest-path distances in the extended graph $(V \cup \{w\}, E \cup (V \times \{w\}))$ with edge lengths 2 on E and 3 on all new edges. See in Figure 2

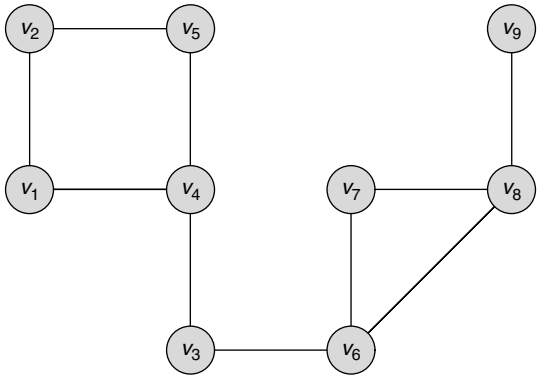

Figure 1 **An Example of Graph** (V, E)

the extended graph for the graph (V, E) of Figure 1, where edges of length 2 are plotted as continuous lines, and the edges of length 3 (those adjacent to w) are dashed.

Then for any set of prototypes $S \subseteq R$,

- w is always correctly classified because I_2 is a singleton.
- For any $v \in I_1 = V$ only one of the following three possibilities for assignment may arise:

—if $v \in S$, it is assigned to itself since dissimilarity is then 0, while $d(v, w) > 0 \ \forall v \neq w$

—if $v \notin S$ but adjacent to some $v' \in S \cap V$, it will be assigned to v' since $d(v, v') = 2$, which is the minimal nonzero possibility for the dissimilarity

—if $v \notin S$ and not adjacent to $S \cap V$, it will be assigned to w since $d(v, w) = 3$, while the dissimilarity to any prototype in V is at least 4.

Thus, misclassification by S of some object is equivalent to being vertex-dominated by $\{w\}$.

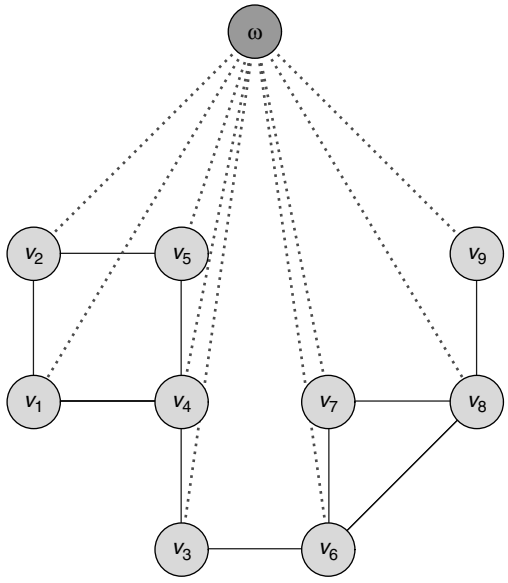

Figure 2 **Extended Graph for the Graph** (V, E) **of Figure 1**

For instance, see Figure 1 with $S = \{w, v_4, v_5, v_6\}$ as set of prototypes. Since $v_5 \in S$, $d(v_5, v_5) = 0 < 3 = d(v_5, w)$; hence, v_5 is assigned to class I_1 so is correctly classified. On the other hand, since v_1 is adjacent to $v_4 \in S$, $d(v_1, v_4) = 2 < 3 = d(v_1, w)$ and v_1 is also correctly classified. However, v_9 is misclassified since $d(v_9, v_4) = 6 > 3 = d(v_9, w)$, $d(v_9, v_5) = 6 > 3 = d(v_9, w)$, and $d(v_9, v_6) = 4 > 3 = d(v_9, w)$, so v_9 is assigned to I_2. Hence, such S does not classify correctly all objects, or, equivalently, it is not a 4-dominating vertex set for the extended graph.

Hence, $S \cap V$ being an l-dominating set is equivalent to all objects being correctly classified by S with $|S| \leq l+1$. □

Corollary 2. *Finding a p-PNN rule is $\mathcal{NP}$-hard.*

In Section 3 we formulate as integer programs the problem of determining the classification rule φ_S with minimal empirical misclassification cost, measured as (8), both for general misclassification penalties and also for the uniform case (5).

3. Integer-Programming Formulations

3.1. General Costs

For each $i \in I$, $s \in R$, let R_{is} denote the set of prototypes that are more preferred by i (according to $\prec_i$) than s, $R_{is} = \{t \in R: t \prec_i s\}$. Let

- $x_s \in \{0, 1\}$, $\forall s \in R$, answering the question "Is candidate s chosen to be a prototype?" Hence, a classification rule φ_S, as defined in (4), is identified by the vector $x \in \{0, 1\}^{|R|}$ defined by

$$x_s = \begin{cases} 1 & \text{if } s \in S \\ 0 & \text{otherwise.} \end{cases}$$

- $y_{is} \in [0, 1]$, $\forall i \in I$ and $s \in R$, answering the assignment question "Is s the prototype least dissimilar to i?" Although, in principle, these variables should be binary, the model allows us to consider them as continuously relaxed to lie between 0 and 1.

The misclassification cost for object $i \in I$ belonging to the class $c(i)$ is then $\sum_{s \in R} r(c(s) \mid c(i)) y_{is}$, so the empirical misclassification cost $\pi_l(S)$ of classification rule φ_S, as defined in (4) is $\sum_{i \in I} \sum_{s \in R} r(c(s) \mid c(i)) y_{is}$.

The problem is to find the set $S \subset R$ with cardinality $|S| = p$ (with p given) containing at least one element in each class c, such that the empirical misclassification cost is minimized. This yields the following (mixed) integer program:

$$\text{(P1)} \quad \min \sum_{i \in I} \sum_{s \in R} r(c(s) \mid c(i)) y_{is}$$

$$\text{subject to} \sum_{s \in R_c} x_s \geq 1 \quad \forall c \in C \tag{9}$$

$$\sum_{s \in R} x_s = p \tag{10}$$

$$\sum_{s \in R} y_{is} = 1 \quad \forall i \in I \tag{11}$$

$$x_s - y_{is} \leq \sum_{t \in R_{is}} x_t \quad \forall (i, s) \in I \times R \tag{12}$$

$$y_{is} \leq x_s \quad \forall (i, s) \in I \times R \tag{13}$$

$$x_s \in \{0, 1\} \quad \forall s \in R$$

$$y_{is} \in [0, 1] \quad \forall (i, s) \in I \times R.$$

Constraints (9) force each class to have at least one prototype and (10) says that we choose p prototypes in total. (11) and (13) ensure that each object has a prototype and that an object can only be assigned to a candidate chosen as prototype. Following Plastria (2002), (12) ensure that an object is assigned to the least dissimilar of the prototypes in the following way:

If s is a prototype and there is no prototype preferred by i **then** i must be assigned to s or equivalently

If $x_s = 1$ and $x_t = 0$, $\forall t \in R_{is}$, **then** $y_{is} = 1$,

which, from Plastria (2002), is expressed by $1 - y_{is} \leq \sum_{t \in R_{is}} x_t + (1 - x_s)$ $\forall (i, s) \in I \times R$, yielding (12).

3.2. Uniform Costs

Now we discuss in some detail the particular case in which the cost structure is given by (5), because, as shown below, we can derive an alternative formulation with fewer variables and constraints than (P1). The additional advantage of this formulation is, remarkably, that its LP relaxation is at least as tight as the LP relaxation of (P1).

Indeed, in this case, the objective of (P1) is

$$\begin{aligned} &\sum_{i \in I} \sum_{s \in R} r(c(s) \mid c(i)) y_{is} \\ &\quad = \sum_{i \in I} \sum_{s \in R_{c(i)}} r(c(s) \mid c(i)) y_{is} + \sum_{i \in I} \sum_{s \notin R_{c(i)}} r(c(s) \mid c(i)) y_{is} \\ &\quad = 0 + \sum_{i \in I} \sum_{s \notin R_{c(i)}} r(c(s) \mid c(i)) y_{is} \\ &\quad = \sum_{i \in I} r_{c(i)} \sum_{s \in R} y_{is} - \sum_{i \in I} r_{c(i)} \sum_{s \in R_{c(i)}} y_{is} \\ &\quad = \sum_{i \in I} r_{c(i)} - \sum_{i \in I} r_{c(i)} \sum_{s \in R_{c(i)}} y_{is}. \end{aligned}$$

Define, for each $i \in I$, $z_i = \sum_{s \in R_{c(i)}} y_{is}$ which answers the (fuzzy) question "Is object i correctly classified?" With this, the objective at any feasible solution is $\sum_{i \in I} r_{c(i)} - \sum_{i \in I} r_{c(i)} z_i$.

To guarantee that the variables z_i answer the question above, the only thing that must be specified is when z_i must be 0: when the choice of z_i's value is left

free there will always be an optimal solution where $z_i = 1$. The former is obtained by stating

If $t \notin R_{c(i)}$ is chosen as a prototype and no prototype in $R_{c(i)}$ is preferred (according to $\prec_i$) than t, **then** $i \in I$ will not be correctly classified
or, for each $t \notin R_{c(i)}$,

If $x_t = 1$ and $x_s = 0$ $(\forall s \in R_{c(i)} \cap R_{it})$, **then** $z_i = 0$, which is expressed by the constraint

$$z_i \le (1 - x_t) + \sum_{s \in R_{c(i)} \cap R_{it}} x_s. \tag{14}$$

Note that this constraint indeed expresses exactly the desired property even when z_i is continuously relaxed (Plastria 2002). Thus, we may rewrite (P1) as

$$\begin{aligned}
\text{(P2)} \quad \min \ & \sum_{i \in I} r_{c(i)} - \sum_{i \in I} r_{c(i)} z_i \\
\text{subject to} \ & \sum_{s \in R_c} x_s \ge 1 \quad \forall c \in C \\
& \sum_{s \in S} x_s = p \\
& z_i \le (1 - x_t) + \sum_{s \in R_{c(i)} \cap R_{it}} x_s \quad \forall i \in I,\ t \notin R_{c(i)} \\
& x_s \in \{0, 1\} \quad \forall s \in R \\
& z_i \in [0, 1] \quad \forall i \in I.
\end{aligned}$$

Calling (LP1) and (LP2) the LP relaxations of (P1) and (P2), we may state

Proposition 3. (LP2) *is at least as tight as* (LP1).

Proof. See Carrizosa et al. (2005). □

Since the integer variables are the same in both models, the subproblems generated when fixing some of these variables satisfy the same property. Therefore, if available, (P2) is preferred when solving the problem with a branch-and-bound algorithm.

Note that $\sum_{i \in I} r_{c(i)}$ in (P2)'s objective is constant, and, since we are only interested in optimal solutions, and not in the optimal objective value, it may be dropped. Sign inversion then leads to the simpler objective $\max \sum_{i \in I} r_{c(i)} z_i$, defining our model (P2′) subject to the same constraints as (P2) above.

The variables above can also be used to model as IPs other variants of NN. For instance, finding the consistent subset of minimal cardinality, i.e., Hart's CNN-rule (Hart 1968), amounts to solving

$$\begin{aligned}
\text{(PCNN)} \quad \min \ & \sum_{s \in S} x_s \\
\text{subject to} \ & \sum_{s \in R_c} x_s \ge 1 \quad \forall c \in C \\
& 1 \le (1 - x_t) + \sum_{s \in R_{c(i)} \cap R_{it}} x_s \quad \forall i \in I,\ t \notin R_{c(i)} \\
& x_s \in \{0, 1\} \quad \forall s \in R.
\end{aligned}$$

Table 1 Parameters of the Databases

Database J	$\lvert J \rvert$	$\lvert C \rvert$	n
glass	214	6	9
glassw	163	3	9
wine	178	3	13
yeastME	258	3	8
abalone	4,177	3	7
spam	4,601	2	57

4. Computational Experience

4.1. Aims

The storage requirements and processing time of the p-PNN rule are smaller than those of NN. Our aim here is to compare empirically the classification power of the p-PNN rule, for different values of p, against NN.

For completeness, we also compare p-PNN with other benchmark methods. In particular, we tested the performance of p-PNN against

- k-NN (k-NN in the tables) for different values of k.
- Support vector machines (SVM), Cristianini and Shawe-Taylor (2000), with linear kernel (Lin), polynomial kernel (Pol), and radial bases function kernel (Rbf).
- Classification trees, denoted here by Trees, (Breiman et al. 1984), with and without pruning (Pruned and Crude respectively), as implemented in Matlab 6.5 Statistics Toolbox.

We performed numerical tests on different standard databases from the UCI Machine Learning Repository (Blake and Merz 1998). The details are in Section 4.2.

Since some of the benchmark methods do not accommodate in a simple way different misclassification

Table 2 Results with k-NN, SVM, and Classification Trees

	k-NN					SVM			Trees	
Database	1-NN	2-NN	3-NN	4-NN	5-NN	Lin	Pol	Rbf	Pruned	Crude
glass										
tr	100	100	100	100	100	60	72	67	78	89
test	71	67	66	66	63	58	62	60	67	64
glassw										
tr	100	100	100	100	100	61	73	71	80	91
test	67	63	67	70	71	56	67	68	71	69
wine										
tr	100	100	100	100	100	99	100	99	96	98
test	95	96	96	95	95	99	96	99	87	90
yeastME										
tr	100	100	100	100	100	87	90	89	90	95
test	80	84	86	87	84	84	86	86	87	88
abalone										
tr	100	100	100	100	100	64	63	65	65	87
test	57	57	60	61	61	63	63	64	63	59
spam										
tr	100	100	100	100	100	90	69	93	94	98
test	91	89	90	90	90	90	69	93	92	92

Table 3 Results of glass

	*p*PNN (%)		Random choice (%)		Heuristic		
p	tr	test	tr	test	tr (%)	test (%)	Time (sec.)
6	66	60	39	32	67	60	1.34
10	70	57	43	31	76	65	1.20
15	76	60	49	42	78	64	1.34
20	80	67	57	49	81	61	1.48

costs, the comparisons were made using the binary cost structure (6). Hence, both (P1) and (P2) apply. Since the LP bound of (P2) is at least as good as that from (P1), by Proposition 3, all our results refer to the simpler variant (P2′) of (P2).

4.2. The Databases

The UCI-Repository databases we used are of different sizes. A first group of databases consists of the Glass Identification Database (called here glass), a subset of the glass database (glassw) consisting only of the "window glass" classes, the Wine Recognition Database (wine), and Yeast Database (yeastME), from which only the three "membrane protein" classes (denoted as ME1, ME2, ME3 in the UCI Repository) are used. Moreover, two bigger databases are also considered: the Abalone Database (abalone), and the Spambase Database (spam). In abalone, three classes (grouping classes 1–8, 9–10, and 11 on) are considered, as cited in Blake and Merz (1998), and the qualitative variable was excluded.

For each database J, the total number of objects $|J|$, the number of classes $|C|$, and the number of variables (all quantitative) n are in Table 1.

The databases contain only continuous variables, and one can thus calculate dissimilarities according to the weighted Euclidean distance, defined for $u = (u_1, \ldots, u_n) \in \mathbb{R}^n$, $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ as

$$d(u, v) = \left(\sum_{1 \le j \le n} \omega_j (u_j - v_j)^2 \right)^{1/2}, \tag{15}$$

Table 4 Results of glassw

	*p*PNN (%)		Random choice (%)		Heuristic		
p	tr	test	tr	test	tr (%)	test (%)	Time (sec.)
3	67	59	37	31	67	58	0.90
5	72	69	44	40	74	65	0.93
10	80	69	51	44	79	71	1.02
15	83	72	60	52	82	74	1.10
20	87	66	63	54	84	70	1.17
30	92	68	70	65	88	67	1.29
40	96	61	74	65	90	63	1.42

Table 5 Results of wine

	*p*PNN (%)		Random choice (%)		Heuristic		
p	tr	test	tr	test	tr (%)	test (%)	Time (sec.)
3	99	99	85	84	98	96	0.91
05	100	93	88	85	99	96	0.97
10	100	94	90	88	100	94	1.10
15	100	97	93	92	100	96	1.19
20	100	95	93	92	100	96	1.26
30	100	95	94	92	100	91	1.38
40	100	97	95	92	100	95	1.52

with each weight ω_i given by

$$\omega_i = \frac{1}{(\bar{\Delta}_i - \underline{\Delta}_i)^2}, \tag{16}$$

where $\bar{\Delta}_i$ and $\underline{\Delta}_i$ represent the highest and the lowest value for the ith variable in the database, respectively. This model is equivalent to considering the unweighted Euclidean distance after rescaling each variable to the interval $[0, 1]$.

All results presented were obtained by ten-fold crossvalidation (Kohavi 1995).

The averaged percentages of correctly classified objects in both the training samples (tr) and testing samples (test) obtained using the benchmark methods are in Table 2.

4.3. Solving to Optimality

The worst-case complexity of the problem was addressed in Section 2.3, and now we discuss the empirical behavior. To compare the running times and the classification power of p-PNN, the IPs were solved on a 2.86 GHz Pentium 4 and 256 MB RAM. CPLEX 8.1.0 was used as the MIP solver. Due to the hardness of these MIP formulations we imposed an upper bound (MAXT) on the computing time of 10,800 seconds. In most instances, running times exceeded MAXT and the optimization process stopped prematurely.

Tables 3–6 display for different values of the number p of prototypes (first column), the percentage of

Table 6 Results of yeastME

	*p*PNN (%)		Random choice (%)		Heuristic		
p	tr	test	tr	test	tr (%)	test (%)	Time (sec.)
3	82	69	70	71	88	84	1.00
5	88	83	74	76	90	86	1.10
10	91	88	77	78	90	86	1.32
15	93	84	77	74	91	81	1.53
20	93	82	78	74	92	85	1.75
30	96	78	80	78	92	82	2.03
40	99	82	81	76	94	82	2.27

Table 7 Results for abalone

	Random choice (%)		Heuristic		
p	tr	test	tr (%)	test (%)	Time (sec.)
3	41	42	63	63	23.40
5	42	42	64	64	25.01
10	48	48	65	64	29.35
30	51	52	66	63	47.02
50	53	53	67	64	64.45
70	54	53	67	63	85.97
500	62	55	70	60	407.40

correctly classified objects in the training sample (second column) and in the testing sample (third column) for the four smallest databases. (The other columns are for later use in Section 4.4.) Further computational results can be found in Carrizosa et al. (2005).

Comparing with the benchmark results in Table 2, several conclusions can be drawn. First, no method systematically outperforms the others. In particular, the p-PNN shows to be comparable against the remaining methods. Moreover, an adequate choice of the parameter p makes our method be among the best classifiers. However, how to choose p is not evident to us, and a crossvalidation process seems to be needed unless the choice of p is guided by the query times requirements.

On the other hand, the computing times are, in all cases, extremely large, suggesting heuristic procedures for solving (P2); see Section 4.4. Moreover, from the columns giving the proportion of correctly classified objects in the training and the testing samples, the former strongly overestimates the latter. Hence, overfitting happens.

4.4. Heuristic Approach

The $\mathcal{NP}$-hardness of the problem as well as the empirical results of Section 4.3 suggest heuristic procedures to speed up computing times. A first and simple choice, yielding promising results (Pekalska et al. 2006), might consist of randomly selecting p prototypes. The results (random choice in Tables 3–8) are

Table 8 Results for spam

	Random choice (%)		Heuristic		
p	tr	test	tr (%)	test (%)	Time (sec.)
2	61	60	84	83	78.67
5	68	67	85	85	80.66
10	69	69	86	85	88.62
25	72	73	88	87	105.57
50	75	75	89	87	122.10
100	78	77	89	87	175.73
500	85	82	92	88	496.95

1. *Initialization.* Randomly choose an initial solution x. Choose a stopping criterion.
2. Repeat until the stopping condition:
   (a) Set $\ell \leftarrow 1$.
   (b) Repeat until $\ell = p$:
      i. Generate randomly a new solution x' differing in at most ℓ prototypes with the current solution x.
      ii. If x' is better than x, set $x' \leftarrow x$ and go to 2(a); otherwise, set $\ell \leftarrow \ell + 1$.
3. Return the best solution found so far.

Figure 3 VNS Heuristic

discouraging, particularly when p is small. Hence, more sophisticated heuristics are needed.

The structure of the problem is such that several existing (meta) heuristic procedures can be easily adapted to our problem. Multistart, genetic algorithms or tabu search have been proposed for prototype selection, with encouraging results (Bezdek and Kuncheva 2001).

VNS combines local search with redefinitions of the neighborhood structure. We use the same neighborhoods as Hansen and Mladenović (1998, 2001a) for the p-median problem. Given a feasible solution, i.e. a set of p prototypes including at least one for each class, its neighborhood of order ℓ consists of all feasible solutions that differ from it in at most ℓ prototypes. The procedure works as described in Figure 3.

In our experiments, the procedure stops after 5,000 calls to step 2(b)i in Figure 3. The results for the small data sets for which the exact solution was also sought with CPLEX are under Heuristic in Tables 3–6. Moreover, much larger data sets, such as abalone or spam, can be handled (Tables 7 and 8).

A very simple heuristic yields, with very low computing times, rather sharp solutions on the training samples. However the quality of the procedures should not be measured on the training sample (which overestimates the quality results) but on the testing sample. On testing samples, the heuristic yields (at much lower computing times) solutions with comparable quality than those obtained with the exact method.

We selected variable neighborhood search (VNS), proposed by Hansen and Mladenović (2001b), both

Table 9 Results with Missing Values for glass

	Fraction of missing values										
p	0	0.025	0.05	0.075	0.1	0.15	0.175	0.2	0.25	0.3	Slope
6	65	61	59	56	58	53	52	56	53	58	−25.70
8	67	60	57	57	55	56	55	54	52	48	−44.84
10	60	61	59	55	57	56	54	55	54	49	−32.09
12	61	58	58	58	55	55	55	53	52	51	−30.34
14	60	60	57	58	59	55	54	55	53	52	−27.56
16	53	58	58	57	56	54	56	55	53	52	−13.12
18	55	59	58	59	56	56	55	55	52	51	−21.90

Table 10 Results with Missing Values for `glassw`

	Fraction of missing values										
p	0	0.025	0.05	0.075	0.1	0.15	0.175	0.2	0.25	0.3	Slope
3	49	59	57	53	58	58	59	54	51	47	−16.19
5	57	61	59	59	60	59	54	57	59	59	−3.72
7	67	61	63	61	59	61	58	59	55	51	−39.83
9	65	64	61	60	60	62	59	59	60	56	−21.62

for simplicity of its implementation and the excellent results obtained for related problems, such as the p-median problem. Other metaheuristics could be used to tackle the MIP (P1), but an empirical comparison of such methods is beyond the scope of this paper.

4.5. Missing Values

As mentioned in Section 2.1, dissimilarities can also be constructed for databases with missing values. We performed experiments to explore the stability of the classification rule with respect to the existence of (many) missing values. For different values of ϑ, a fraction ϑ of data were randomly chosen and replaced by nulls.

We considered the dissimilarity described in (3), with each ω_j defined by (16). The optimization was performed using the VNS heuristic from Section 4.4 with its stopping rule. To reduce the random effects due to the inclusion of nulls, we ran each test 100 times. The average proportion of correctly classified objects in the testing sample, for different values of p and ϑ, together with the slopes of the regression lines linking percentage of correctly classified objects with fraction of missing data, are shown in Tables 9–12 for the small databases; see Carrizosa et al. (2005) for further details.

As expected, the quality of the classification rule deteriorates as the number of nulls increases. However, the correct classification rates decrease slowly since, for instance, in `yeastME`, the rules still classify correctly more than 70% of the objects when 30% of the values are missing. It is not evident how the degradation in classification is affected by the number of prototypes, since no trend is found in the slopes of the corresponding regression lines.

Table 11 Results with Missing Values for `wine`

	Fraction of missing values										
p	0	0.025	0.05	0.075	0.1	0.15	0.175	0.2	0.25	0.3	Slope
3	92	94	93	94	93	92	91	85	92	90	−14.58
5	96	95	90	95	93	93	92	93	89	88	−20.61
7	96	94	95	94	93	94	93	92	90	89	−20.27
9	94	94	93	93	93	92	90	88	90	88	−21.53

Table 12 Results with Missing Values for `yeastME`

	Fraction of missing values										
p	0	0.025	0.05	0.075	0.1	0.15	0.175	0.2	0.25	0.3	Slope
3	82	78	76	74	79	77	76	74	73	71	−25.90
5	77	79	81	79	75	75	75	74	70	71	−30.91
7	84	79	79	75	78	75	76	75	73	71	−32.79
9	78	74	79	76	76	74	74	71	72	73	−19.45

5. Conclusion and Further Research

We have introduced an optimization-based method for multiclass classification problems. Since the only requirement for the data is knowledge of a dissimilarity between entries, no statistical assumptions on data are needed, and qualitative variables, as well as missing values, are easily handled. Contrary to other competitive procedures, differences in misclassification costs are naturally accommodated within the model.

MIP formulations have been developed yielding classifiers with performance comparable to benchmark procedures. However, the computational effort required makes them prohibitive for databases of moderate size. Stronger MIP formulations, using valid inequalities, as well as more sophisticated bounding strategies, should be investigated since they might allow solving larger instances of this $\mathcal{NP}$-hard problem in reasonable time.

For large databases heuristics seem to be the only feasible approach. The results obtained using VNS are encouraging. An empirical comparison with other (meta)heuristics remains.

The dissimilarity d has been considered given. However, the dissimilarity itself can be seen as a (modelling) decision variable. Hints to choose an appropriate d, e.g. by choosing appropriate weights ω_j in (15), are now under research.

Further study is also needed for the choice of the parameter p. Indeed, our numerical results do not lead to clear guidelines for choosing p. We propose crossvalidation, but in applications in which querying time is a critical issue, p will have to be fixed exogenously.

Acknowledgments

The authors thank two anonymous referees for their remarks, which have improved the quality of this paper. This research is supported by Grant MTM2005-09362-C03-01 of Ministerio de Educación y Ciencia, Spain.

References

Altschul, S. F., M. S. Boguski, W. Gish, J. C. Wootton. 1994. Issues in searching molecular sequence databases. *Nature Genetics* **6** 119–129.

Altschul, S. F., W. Gish, W. Miller, E. W. Myers, D. J. Lipman. 1990. Basic local alignment search tool. *J. Molecular Biol.* **215** 403–410.

Bennett, M. V., T. R. Willemain. 2004. The filtered nearest neighbor method for generating low-discrepancy sequences. *INFORMS J. Comput.* **16** 68–72.

Bezdek, J. C., L. I. Kuncheva. 2001. Nearest prototype classifier designs: An experimental study. *Internat. J. Intelligent Systems* **16** 1445–1473.

Blake, C. L., C. J. Merz. 1998. UCI Repository of Machine Learning Databases. Department of Information and Computer Sciences, University of California, Irvine, Irvine, CA, http://www.ics.uci.edu/~mlearn/MLRepository.html.

Breiman, L., J. H. Friedmann, R. A. Olshen, C. J. Stone. 1984. *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Brighton, H., C. Mellish. 2002. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery* **6** 153–172.

Carrizosa, E., B. Martin-Barragan, F. Plastria, D. Romero Morales. 2005. A dissimilarity-based approach for classification. Technical report, METEOR Research Memorandum RM/02/027, University of Maastricht, The Netherlands.

Cochran, W. G. 1977. *Sampling Techniques*, 3rd ed. Wiley, New York.

Cover, T. M., P. E. Hart. 1967. Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory* **13** 21–27.

Cristianini, N., J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK.

Dasarathy, B. V. 1991. *Nearest Neighbor* (*NN*) *Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, CA.

Devroye, L., L. Györfi, G. Lugosi. 1996. *A Probabilistic Theory of Pattern Recognition*. Springer, New York.

Freed, N., F. Glover. 1981. Simple but powerful goal programming models for discriminant problems. *Eur. J. Oper. Res.* **7** 44–60.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York.

Gehrlein, W. V. 1986. General mathematical programming formulations for the statistical classification problem. *Oper. Res. Lett.* **5** 299–304.

Geva, S., J. Sitte. 1991. Adaptive nearest neighbor pattern classifier. *IEEE Trans. Neural Networks* **2** 318–322.

Gochet, W., A. Stam, V. Srinivasan, S. X. Chen. 1997. Multigroup discriminant analysis using linear programming. *Oper. Res.* **45** 213–225.

Hansen, P., N. Mladenović. 1998. Variable neighborhood search for the *p*-median. *Location Sci.* **5** 207–226.

Hansen, P., N. Mladenović. 2001a. Variable neighborhood decomposition search. *J. Heuristics* **7** 335–350.

Hansen, P., N. Mladenović. 2001b. Variable neighborhood search: Principles and applications. *Eur. J. Oper. Res.* **130** 449–467.

Hart, P. E. 1968. The condensed nearest neighbor rule. *IEEE Trans. Inform. Theory* **14** 515–516.

Hastie, T., R. Tibshirani, J. Friedman. 2001. *The Elements of Statistical Learning*. Springer, New York.

Kaufman, L., P. J. Rousseeuw. 1990. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, New York.

King, R. D., C. Feng, A. Sutherland. 1995. Statlog: Comparison of classification algorithm in large real-world problems. *Appl. Artificial Intelligence* **9** 289–333.

Kohavi, R. 1995. Cross-validation and bootstrap for accuracy estimation and model selection. *Proc. 14th Internat. Joint Conf. Artificial Intelligence*. Morgan Kaufmann, San Fransisco, CA, 1137–1143.

Kuncheva, L. I. 1997. Fitness function in editing *k*-NN reference set by genetic algorithms. *Pattern Recognition* **30** 1041–1049.

Kuncheva, L. I., J. C. Bezdek. 1998. Nearest prototype classification: Clustering, genetic algorithm or random search? *IEEE Trans. Systems, Man, and Cybernetics, Part C* **28** 160–164.

Lipowezky, U. 1998. Selection of the optimal prototype subset for 1-NN classification. *Pattern Recognition Lett.* **19** 907–918.

Mangasarian, O. L. 1994. Misclassification minimization. *J. Global Optim.* **5** 309–323.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, New York.

Pearson, W. R., D. J. Lipman. 1988. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci.* **85** 2444–2448.

Pekalska, E., R. P. W. Duin, P. Paclík. 2006. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition* **39** 189–208.

Plastria, F. 1995. Continuous location problems. Z. Drezner, ed. *Facility Location. A Survey of Applications and Methods*. Springer-Verlag, New York, 229–266.

Plastria, F. 2001. Asymmetric distances, semidirected networks and majority in Fermat-Weber problems. *Locator: ePublication of Location Analysis* **2** 15–62.

Plastria, F. 2002. Formulating logical implications in combinatorial optimisation. *Eur. J. Oper. Res.* **140** 338–353.

Thompson, S. K. 2002. *Sampling*. Wiley, New York.

Yang, M.-S., H.-M. Shih. 2001. Cluster analysis based on fuzzy relations. *Fuzzy Sets and Systems* **120** 197–212.

Zimmermann, H. J. 1991. *Fuzzy Set Theory and Its Applications*. Kluwer, Dordrecht, The Netherlands.